RELATÓRIO TÉCNICO

UM PROTÓTIPO COMPLETO DE UM SISTEMA DE INFORMAÇÃO ORIENTADO A OBJETOS EM SMALLTALK

Miguel Jonathan

NCE-22/90
Outubro/90

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
BRASIL

Este artigo foi publicado originalmente nos Anais do XXIII Congresso Nacional de Informática da SUCESU Rio de Janeiro 1990.

UM PROTOTIPO COMPLETO DE UM SISTEMA DE INFORMAÇÃO ORIENTADO A OBJETOS EM SMALLTALK

RESUMO:

O trabalho apresenta um sistema completo de informação em Smalltalk, utilizando o paradigma de orientação a objetos. O sistema é utilizado para o controle operacional de uma biblioteca típica. Os aspectos essenciais desse paradigma de programação sao examinados, especialmente encapsulação de dados e programas, re-utilização de objetos pré-definidos, hierarquia de classes, polimorfismo e modularização automática.

A COMPLETE OBJECT-ORIENTED INFORMATION SYSTEM PROTOTYPE IN SMALLTALK

ABSTRACT:

This work presents a complete information system written in Smalltalk using the object-oriented paradigm. The system is designed as a tool for the operational control of a typical library. The essential aspects of this programming paradigm are examined, mainly data and programs encapsulation, re-usability of pre-defined objects, class hierarchy, polymorphism and automatic modularization.

UM PROTÓTIPO COMPLETO DE UM SISTEMA DE INFORMAÇÃO ORIENTADO A OBJETOS EM SMALLTALK

AUTOR:
MIGUEL JONATHAN

ENDEREÇO:
Núcleo de Computação Eletrônica, UFRJ
Departamento de Ciência da Computação, IM-UFRJ
Caixa Postal 2324
CEP 20001 - RIO DE JANEIRO - RJ
Tel: (021)290-3212
e-mail: ncs01002@ufrj.bitnet

CURRICULUM VITAE:
Engenheiro Eletrônico (ITA), Mestre em Engenharia Elétrica (PUC/RJ), foi Analista de Sistemas do SERPRO e Professor da COPPE/UFRJ. Atualmente é Analista do NCE/UFRJ e Professor Adjunto do Departamento de Ciência de Computação do Instituto de Matemática da UFRJ.

TOTAL DE PÁGINAS: 13

RESUMO:

O trabalho apresenta um sistema completo de informação em Smalltalk, utilizando o paradigma de orientação a objetos. O sistema é utilizado para o controle operacional de uma biblioteca típica. Os aspectos essenciais desse paradigma de programação são examinados, especialmente encapsulação de dados e programas, re-utilização de objetos pré-definidos, hierarquia de classes, polimorfismo e modularização automática.

PALAVRAS-CHAVE:
Orientação para Objetos, Sistemas de Informação, Smalltalk.

1. INTRODUÇÃO

A concepção de um sistema de informação é profundamente influenciada pela linguagem de programação utilizada. A linguagem delimita em boa parte os conceitos e estruturas com os quais podemos trabalhar, e induz a soluções que melhor se adaptam aos recursos de expressão de que dispõe.

A introdução recente de linguagens orientadas para objetos, das quais Smalltalk [2] foi pioneira, abriu caminho para se reconsiderar a forma tradicional de se conceber e construir sistemas de informação computarizados. Até então, o uso de linguagens algorítmicas convencionais como COBOL ou PL/1, levaram a uma separação aparentemente natural entre programas e dados. Um sistema é composto por um conjunto de programas e arquivos de dados. Para operar o sistema, o operador aciona determinado programa, que contém informação sobre o nome de um ou mais arquivos lógicos de onde lerá e para onde gravará dados. A ligação entre os arquivos físicos e lógicos é feita por comandos de contrôle do sistema operacional ou pelo gerenciador de banco de dados. Nada em princípio impede que o programa acesse o arquivo errado, ou que outro programa acesse o mesmo arquivo para a mesma finalidade.

Na programação orientada para objetos deixa de existir essa separação entre programas e dados. Um sistema é concebido como uma interação de objetos que trocam *mensagens* entre si. Cada objeto pertence a uma determinada *classe*, e cada classe contém a descrição da *estrutura de dados* bem como da *funcionalidade* associada a todas as *instâncias* (objetos) da classe. Por funcionalidade entende-se a *reação* dos objetos a cada mensagem contida no protocolo da classe, e que é implementada através de rotinas específicas chamadas *métodos*.

Um objeto só reage a mensagens que estejam contidas no protocolo de sua classe. Dessa forma, torna-se impossível alterar ou examinar os dados de um objeto sem utilizar as mensagens apropriadas, o que aumenta o grau de segurança. Por outro lado, mudanças no sistema podem ser introduzidas de forma localizada, através de modificações em determinadas classes, sem risco de afetar outras partes do sistema, desde que não se altere o protocolo original, o que aumenta a manutenibilidade, extensibilidade e confiabilidade do sistema.

O encapsulamento de características dos objetos em classes tem como seu mais importante sub-produto a re-usabilidade de "design" e de código. Uma vez que uma classe foi idealizada e implementada, ela pode ser utilizada pelas mais diversas aplicações sem necessidade de se re-codificar, re-testar ou re-documentar. Um sistema pode então ser projetado aproveitando muitas classes pré-fabricadas como "componentes", o que possibilita reduzir a complexidade do projeto, com grandes ganhos de produtividade.

Classes são organizadas em hierarquias de generalização / especialização, permitindo ao projetista modelar seus objetos de forma semelhante à utilizada pela mente humana na classificação dos objetos da realidade. Assim, propriedades comuns a várias sub-classes podem ser localizadas em uma única classe, acima

na hierarquia, reduzindo ainda mais o número de linhas de código, e evitando inconsistências no sistema.

Smalltalk é um ambiente e uma linguagem inteiramente orientados para objetos. Possui uma extensa biblioteca de classes pré-fabricadas e de uso geral, entre as quais estão janelas, menus, e diversas classes de coleções (conjuntos, vetores, etc.). O sistema descrito neste trabalho foi totalmente escrito em Smalltalk/V [3] e utiliza largamente os conceitos acima mencionados. Outras linguagens orientadas para objetos, como C++ [6], Turbo Pascal 5.5 [7] e Eiffel [9] podem ser utilizadas com igual sucesso, bastando que disponham de bibliotecas adequadas.

2. CONCEPÇÃO DO SISTEMA

O sistema apresentado neste trabalho tem por finalidade a operação de uma biblioteca típica. Ele foi implementado em um computador tipo IBM PC-XT, utilizando Smalltalk/V. Nesta seção são descritas as classes de objetos que foram criadas, e mencionadas as principais classes pré-definidas que foram utilizadas.

2.1 Descrição sumária das classes criadas

As classes criadas especialmente são:

Leitor
Publicação
Exemplar
Empréstimo
Reserva
Biblioteca
SessãoBiblioteca

SessãoBiblioteca é a classe que descreve a interface gráfica do sistema. Cada vez que o usuário abre uma sessão para operar a biblioteca, uma instância dessa classe é criada. A classe provê toda a funcionalidade para operar com janelas múltiplas sincronizadas e menus para ativar funções do sistema. A instância desaparece ao final da sessão.

Biblioteca é a classe que representa o sistema. A uma instância específica dessa classe estão associados os leitores, publicações, exemplares, empréstimos e reservas que de outra forma não teriam um elemento permanente de unificação. O protocolo de Biblioteca contém mensagens para realizar todas as operações necessárias para operá-la. O papel de uma SessãoBiblioteca é na verdade o de estabelecer um diálogo com o usuário, via interface gráfica, e encaminhar mensagens correspondentes à Biblioteca.

Leitor é uma classe *abstrata*, no sentido de que não possui instâncias diretamente. Quem possui instâncias são suas sub-classes, *LeitorAluno* e *LeitorProfessor*. A idéia é que as características comuns a todos os leitores (dados

estruturais: nome, endereço; mensagens e rotinas comuns: emprestar livro, obter endereço, etc.) ficam localizadas na classe Leitor. E as propriedades específicas de cada categoria de leitor ficam localizadas nas respectivas sub-classes. Neste sistema, um LeitorAluno pode retirar livros, mas não periódicos (diferença funcional). Além disso, possui um componente específico, que é o nome do curso no qual está matriculado (diferença estrutural). Por outro lado, um LeitorProfessor pode retirar livros e periódicos, e possui também um componente específico, que é o nome do departamento a que pertence. Os prazos de empréstimo para cada categoria também diferem.

Publicação é outra classe abstrata. Suas sub-classes são *Livro* e *Periódico*, e são elas que possuem instâncias. Na classe Publicação estão descritos os componentes comuns (título, identificador, conjunto de exemplares, fila de reservas) e métodos comuns. A classe Livro descreve os componentes próprios (autor, conjunto de palavras-chave) e a classe Periódico os componentes número, volume, mes e ano.

Exemplar é uma classe criada para diferenciar os vários exemplares de uma mesma Publicação. Cada instância de Exemplar possui um identificador próprio, e uma referência à Publicação correspondente. Outros componentes de Exemplar são o seu estado de disponibilidade para empréstimo (verdadeiro ou falso) e o motivo de sua eventual não-disponibilidade. São os Exemplares que são efetivamente emprestados, e não a Publicação.

Empréstimo é uma classe onde cada instância reprsenta um empréstimo de um exemplar para um leitor. Além de referenciar o exemplar e o leitor, um Empréstimo contém ainda a data do empréstimo e a data limite para devolução.

Reserva é uma classe onde cada instância representa uma reserva de uma publicação para um leitor. Note que a reserva é feita para a publicação, e não para algum exemplar específico. Cada vez que uma nova reserva é solicitada, o método correspondente cria uma nova instância de Reserva e a coloca no fim da fila de reservas da publicação. Os componentes de cada Reserva são o leitor (LeitorAluno ou LeitorProfesor), a publicação (Livro ou Periódico) e a data de solicitação.

2.2 Utilização de classes pré-definidas

As classes acima foram idealizadas e criadas especialmente para este sistema. O sistema também utiliza diversas outras classes pré-definidas em Smalltalk/V, algumas das quais são relacionadas a seguir:

Dicionário (Dictionary): um Dicionário é um conjunto de pares argumento / valor, onde o argumento (ou chave) é único. Dicionários são usados neste sistema para relacionar o identificador único de leitores, publicaçòes e exemplares com as referências internas às respectivas instâncias. Outro dicionário é usado para relacionar cada palavra-chave com o conjunto de livros que a contêm. Esses dicionários são componentes da classe Biblioteca. Note que o conceito de dicionário é bastante geral: o valor associado ao argumento pode ser um objeto de qualquer classe.

Conjunto (Set): Smalltalk contém uma extensa variedade de sub-classes da classe (abstrata) *Coleção* (Collection), que representam coleções de objetos com propriedades variadas. Conjunto é uma delas. A sua característica básica é não permitir objetos duplicados. Conjuntos são usados em vários pontos deste sistema: conjunto das palavras-chave de um Livro, conjunto de exemplares de um Livro ou Periódico, etc. Dicionário é uma sub-classe de Conjunto, especializada para representar conjuntos de instâncias da classe Associação, cada uma composta por um par de objetos argumento / valor.

ColeçãoOrdenada (OrderedCollection): é outra sub-classe de Coleção. Sua caracteristica é manter a noção da *ordem* relativa dos objetos contidos na coleção (primeiro, segundo, último). É usada para implementar diretamente estruturas como filas, pilhas e listas. Neste sistema, a fila de reservas de cada publicação é implementada como uma ColeçãoOrdenada. Outra propriedade interessante desta classe é a de se expandir e comprimir automàticamnte à medida que objetos são incluídos ou retirados.

ColeçãoClassificada (SortedCollection): é uma sub-classe de ColeçãoOrdenada, que possui a propriedade adicional de manter os objetos nela contidos classificados de acordo com um critério de ordenação pré-determinado. As listas de nomes de leitores e títulos de publicações, que são apresentadas pela interface do sistema em ordem alfabética, são produzidas fazendo com que sejam geradas por meio de coleções desta classe.

Data (Date): a classe Data provê todo o protocolo e métodos para tratar com datas. Esta classe é usada no sistema para determinar a data de devolução a partir da data do empréstimo e do prazo específico do leitor para o tipo de publicação. É usada também para determinar a ordem relativa entre duas datas, e para imprimir datas em formatos variados.

Janela (Pane): uma das classes fundamentais para possibilitar a interface gráfica, permite a criação de janelas na tela com multiplas sub-divisões. Cada divisão pode ser ser uma de suas sub-classes especializadas para escrever e editar textos em geral *(TextPane)*, para apresentar listas para seleção de um elemento *(ListPane)*, ou ainda para apresentar figuras gráficas *(GraphPane)*.

Menu (Menu): é uma classe para ser usada em associação com a anterior. Permite criar para cada sub-janela um menu específico com as opções de operação, cada uma destinada a ativar um método específico da aplicação.

Diversas outras classes pré-definidas foram usadas, algumas comumente encontradas em outras linguagens, como números, caracteres, cadeias e vetores, e outras específicas para a utilização rotineira de Smalltalk, e que não são aqui mencionadas.

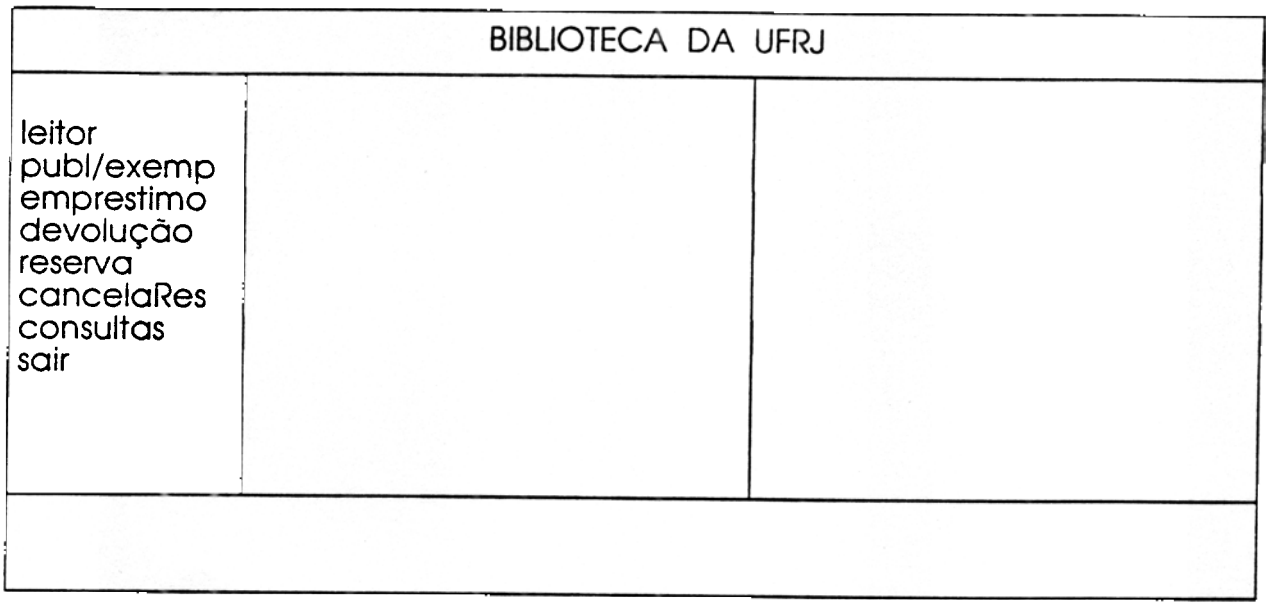
A questão importante a ressaltar, no que se refere à especificidade da metodologia, é o fato do sistema ser idealizado de forma a aproveitar a disponibilidade dessas classes. O papel do analista é modificado, no sentido de que ele dispõe de uma variedade muito maior de componentes pré-fabricados para incorporar ao seu "design". Um paralelo pode ser traçado com o projetista de hardware que utiliza largamente pastilhas pré-fabricadas com funcionalidades específicas. Devido a essa similaridade, a expressão "software chip" foi cunhada

por Cox [1] para representar a funcionalidade incorporada nas classes das linguagens orientadas para objetos.

3. INTERFACE GRÁFICA E FUNÇÕES DO SISTEMA

Esta seção descreve a interface gráfica que se apresenta ao operador do sistema e as funções disponíveis para operar a biblioteca. A interface foi desenvolvida aproveitando as classes pré-definidas para criação de janelas e menus.

A figura abaixo ilustra a aparência da interface, ao iniciar uma sessão:



A janela externa (TopPane) é dividida em 4 sub-janelas ("panes"). A da esquerda é um *ListPane*, usada aqui para exibir as funções de operação do sistema, e que podem ser selecionadas com o "mouse" ou com as teclas de deslocamento e seleção. ListPane's são janelas especializadas para apresentar uma lista de cadeias de caracteres ou símbolos, um em cada linha. Ao selecionar uma linha específica, o seu conteúdo é passado como argumento para uma rotina associada, que pode então decidir o processamento a seguir.

A sub-janela do centro é outro ListPane, que é usado para apresentar listas de nomes de leitores, ou títulos de publicações, em ordem alfabética para seleção.

A sub-janela da direita é um *TextPane*, onde são editadas as entradas de dados e apresentadas as saídas diversas de informaçòes do sistema. TextPane's são janelas especializads em edição de textos, e já incorpora um menu com funções variadas. A função *salvar* ("save") permite recuperar o conteúdo da janela para procesamento posterior.

A sub-janela inferior é outro TextPane, usado pelo sistema para apesentar mensagens diversas de erro, avisos ou orientção para o operador.

Em Smalltalk, cada janela é normalmente associada a um *modêlo*. O modêlo é um objeto em cuja classe estão definidos os métodos para abrir a janela, criar os menus de cada sub-janela, executar as operações associadas aos menus, e processar os valores selecionados das janelas de lista e de texto. Neste sistema, o modêlo da janela da interface é uma instância da classe SessãoBiblioteca. O modêlo é o elo de comunicação entre a janela e os demais objetos do sistema.

Uma SessãoBiblioteca é criada selecionando a opção de mesmo nome no menu principal do sistema Smalltalk, prèviamente alterado para este fim. Ao final da sessão, o operador seleciona a opção *sair* da lista de funções, que provoca o fechamento da janela e a remoção da instância de SessãoBiblioteca.

3.1 Funções do sistema Biblioteca

Esta seção descreve de forma resumida as funções do sistema associadas a cada uma das opções disponíveis na janela da esquerda. O objetivo é proporcionar uma idéia da diversidade de funções, e de como os recursos da interface e das classes pré-definidas foram utilizados na construçào do sistema.

leitor esta opção permite operar com o cadastro de leitores. Um menu adicional solicita a especificação da sub-classe desejada (LeitorAluno ou LeitorProfessor). A relação alfabética dos nomes dos leitores é listada na janela central. As operações possíveis sobre o cadastro são:

a. Incluir novo leitor:
Basta acionar o menu da janela com a lista de nomes e selecionar a opção *incluir*. Os dados completos do novo leitor podem ser então editados na janela da direita, preenchendo o formulário que ali aparece. Após entrar com os dados, o operador ativa a opção *salvar* do menu. Uma nova instância da sub-classe de Leitor é criada, e a lista de leitores é atualizada em seguida, já com o nome do novo leitor na posição correta.

b. Excluir um leitor:
O operador seleciona o nome do leitor na lista, e ativa a opção *excluir* do menu da lista. A instância correspondente é removida do sistema. Note que, ao selecionar o nome do leitor, seus dados completos são dispostos na sub-janela de edição. Caso se queira desfazer a operação, basta acionar em seguida a opção *salvar* do menu desta janela para re-incluir o leitor. A operação de exclusão é

abortada caso haja empréstimos em nome do leitor, e uma mensagem de aviso é colocada na janela inferior.

c. Alterar os dados de um leitor:
Após selecionar o nome do leitor da lista, os seus dados aparecem na janela da direita e podem ser editados à vontade, e re-gravados através da opção *salvar* do menu. Note que não há necessidade de se acionar nenhuma opção de "alterar dados". O sistema permite inclusive alterar o identificador único do leitor. Ao contrário dos sistemas convencionais, o identificador único de um objeto é opcional, e usado apenas para fins externos. Cada instância é identificada internamente por meios não acessíveis ao usuário. O sistema Biblioteca prevê duas possibilidades no caso do operador alterar o valor do identificador: ou o operador deseja realmente trocar o identificador do leitor, ou então deseja incluir um novo leitor (com outro identificador) aproveitando parte dos dados desse leitor que já se encontram na tela. Para isso, ao detetar uma alteração no valor do identificador, um menu adicional é apresentado para que o operador explicite a sua intenção.

publ/exemp esta opção permite operar com o cadastro de publicações e exemplares. A operação é semelhante à descrita para o cadastro de leitores. o operador seleciona a sub-classe desejada (Livro ou Periódico), e a lista de títulos é apresentada em ordem alfabética. Os dados da janela de edição permitem também incluir/excluir exemplares, bem como marcá-los como disponíveis ou não-disponíveis para empréstimo.

O sistema impede a exclusão de uma publicação caso existam exemplares dela emprestados. Os dados de uma Publicação e seus exemplares continuam no sistema após a exclusão. A data de exclusão é registrada como *data da baixa* e o motivo da baixa é solicitado e registrado, para fins de inventário.

empréstimo esta opção é usada para registrar um novo empréstimo ou para prorrogar um empréstimo existente. Após selecionar a opção, os nomes dos leitores são listados em ordem alfabética na janela central. O operador seleciona o leitor desejado e um "prompt" solicita o identificador do exemplar a ser emprestado. "Prompts" são janelas especializadas para entrada de dados solicitados pelo sistema. Após entrar com o identificador do exemplar, o operador é solicitado a confirmar os dados do empréstimo, dispostos na janela de edição. Caso não haja impedimento para a realização do empréstimo, e tratando-se de um novo empréstimo, o sistema cria uma nova instância da classe Empréstimo, e registra nos seus componentes referências ao Leitor e ao Exemplar, a data do empréstimo e a data limite para devolução. As datas são calculadas pelo sistema, e são escritas na janela de edição. O Exemplar é marcado como não-disponível, com motivo "emprestado". O sistema verifica ainda se existe alguma Reserva da publicação para esse mesmo leitor, e caso

exista ela é removida. A instância de Empréstimo criada é incorporada à coleção de empréstimos da Biblioteca.

Caso o exemplar já esteja emprestado para o mesmo leitor, a Biblioteca pede confirmação de que se trata de uma prorrogação, e atualiza a data limite para devolução no Empréstimo correspondente. Após a efetivação do empréstimo ou da prorrogação, o "prompt" solicitando identificador de exemplar é novamente re-apresentado para permitir entrar com outro empréstimo para o mesmo leitor.

Existem situações em que um empréstimo não pode ser completado: exemplar não disponível (reservado, etc.), leitor nào autorizado, e outras. Nesses casos uma mensagem explicativa aparece na janela inferior, e o sistema solicita outro identificador de exemplar, reiniciando o processo. A resposta nula ou brancos remete o sistema para a seleçào de outro leitor.

O método que gera a lista de leitores é o mesmo usado para a opção anterior de cadastro. Essa solução foi adotada para evitar a necessidade de se memorizar números de identificação dos leitores. Já os identificadores de exemplares podem ser facilmente transcritos de suas etiquetas de identificação.

devolução ao selecionar esta opção, um "prompt" solicita o identificador do exemplar que está sendo devolvido. A Biblioteca examina a sua coleção de empréstimos, e procura aquele que contém referência ao exemplar dado. Os dados do exemplar e da publicação são apresentados na janela de edição para confirmação, com informação sobre atraso, se houver. Após a confirmação, a instância de Empréstimo é removida. A Biblioteca envia então uma mensagem ao Exemplar para marcá-lo novamente como disponivel para empréstimo. Ao receber a mensagem, o Exemplar verifica se existe alguma reserva pendente na lista de reservas da sua Publicação, ou seja, se há instâncias de Reserva sem referência a algum Exemplar. Caso exista, o Exemplar envia mensagem à Publicação para associar-se à Reserva da frente da fila, o que resulta em ser marcado como não-disponível com motivo "reservado". Caso não haja reservas, o Exemplar se marca como disponível.

reserva essa opção permite gerar uma Reserva de uma determinada Publicação para um Leitor. Ao ser ativada, a lista de nomes de leitores é apresentada como no caso de empréstimos. Após a seleção do leitor, a lista de publicações é apresentada para seleção. Caso existam exemplares disponíveis da publicação, estes são listados na janela de edição, e um "prompt" solicita do operador que indique o exemplar a reservar. Nesse caso a instância de Reserva é criada com referência direta ao Exemplar. Caso não haja exemplar disponível, a Reserva é criada com referência à Publicação, permanecendo pendente até que um exemplar se torne disponível por qualquer motivo (devolução, aquisição nova, reserva cancelada, etc.). Quando isso ocorrer, a primeira Reserva da fila será associada ao exemplar, e

um aviso enviado ao leitor para retirá-lo até uma data limite. Após a data limite, uma Reserva é automaticamente removida, e o Exemplar torna-se novamente disponível.

cancelaRes esta opção permite cancelar reservas prèviamente registradas. O operador seleciona o nome do leitor da lista, e as reservas em seu nome são listadas no janela de edição, uma a uma, dando opção de cancelamento ou não, através de "prompts".

consultas ao ativar esta opção, um novo menu é apresentado, permitindo ao operador realizar diversas consultas ao sistema. Esse menu, que pode ser expandido, possui atualmente as seguintes opções:

-lista leitores em atraso
-lista livros com determinada palavra-chave
-lista publicações com determinado leitor

Para cada consulta, o sistema solicita parâmetros adicionais, se necessário, e exibe as respostas na janela de edição.

sair encerra a sessão. A janela desaparece e a instância de SessãoBiblioteca é removida.

4. CONSIDERAÇÕES SOBRE A IMPLEMENTAÇÃO

4.1 Limitações do ambiente

Alguns fatores ainda impedem que sistemas como esse tenham uso real efetivo, fatores esses que tenderão a ser reduzidos à medida que as linguagens e ambientes orientados a objetos sejam aperfeiçoados.

Em Smalltalk/V, por exemplo, todos os objetos do sistema permanecem na memória durante a execução. Não há banco de dados em memória secundária, o que limita o número de instâncias possíveis de todos os objetos a 64K. Neste estudo, esse fator não causou limitação, e o tempo de resposta foi bastante satisfatório.

Já existem disponíveis comercialmente gerenciadores de banco de dados orientados a objetos para ambientes similares a Smalltalk [4], e a pesquisa nesse campo tem sido intensa, indicando que esta limitação é temporária.

4.2 Metodologia de desenvolvimento

Não foi utilizada neste sistema nenhuma metodologia estruturada convencional de análise e projeto. O método usado foi o de *desenvolvimento incremental evolutivo* sugerido em [3] para o desenvolvimento de aplicações em Smalltalk: a partir de uma descrição inicial do problema, o analista projeta a versão inicial da interface, e a seguir as classes, componentes e métodos. As facilidades inerentes do ambiente e a modularidade natural do sistema produzido permitem implementar e testar ràpidamente um protótipo. Este exercício permite obter novas percepções sobre a aplicação, produz novas idéias de melhoria, levando a uma nova versão que pode ser gerada também ràpidamente por meio de modificações localizadas na versão anterior. Essa facilidade de produzir e testar versões sucessivamente mais elaboradas estimula a criatividade e a interação com o usuário.

4.3 Concisão de código

A utilização do paradigma de orientação a objetos e a utilização de grande número de classes pré-definidas permitiu uma implementação altamente modular, concisa e facilmente extensível. O número total de linhas de código efetivamente escritas gira em torno de 800, o que é extremamente baixo para um sistema desse porte. Os módulos são em geral pequenos, a maioria com entre 1 e 10 linhas de código.

A concisão de código deve-se também às facilidades da linguagem que permite utilizar métodos poderosos para processar informações contidads em coleções de objetos, simulando comandos normalmente encontrados em linguagens de consulta a bancos de dados.

Por exemplo, no método que implementa a consulta:

"lista leitores em atraso"

a expressão abaixo faz a variável atr conter apenas os empréstimos com atraso:

```
atr := bibl emprestimos select:[:emp | emp dataDevol < Date today].
```

A interpretação da expressão acima é: "selecione da coleção de empréstimos da biblioteca aqueles empréstimos tais que a sua data de devolução seja menor que a data de hoje, e coloque-os em uma outra coleção referenciada pela variável atr".

4.4 Polimorfismo

Linguagens orientadas a objetos, como Smalltalk, são naturalmente polimórficas. A mesma mensagem, enviada a objetos de classes diferentes, pode ativar métodos diferentes, embora com efeitos externos semelhantes. Essa característica permite reduzir grandemente o número de identificadores de mensagens, simplificando a operação e facilitando a extensibilidade das aplicações.

Por exemplo, a mensagem

empresteAPeriodico: umExemplar

quando enviada a um LeitorProfessor, ativa um método que procura realizar o empréstimo do exemplar de periódico dado como argumento da mensagem. Caso o receptor da mesma mensagem seja um LeitorAluno, o método ativado será outro, resultando numa mensagem de aviso de que alunos não podem retirar periódicos.

Neste projeto foi utilizada uma técnica simples, mas eficiente, de polimorfismo múltiplo, descrita por Ingalls em [5], e que permite ampliar a generalidade do nome de uma mensagem de modo a se aplicar não só a receptores, como também a argumentos de classes diversas. No caso específico, a mensagem:

empresteA: umLeitor

pode ser enviada a um objeto de qualquer sub-classe de Publicação (Livro, Periódico), tendo como argumento um objeto de qualquer sub-classe de Leitor (LeitorAluno, LeitorProfessor).

O uso de polimorfismo simplifica o projeto, pois a rotina que solicita o empréstimo não precisa saber de antemão quantos métodos distintos existem (ou existirão) para realizar empréstimos de qualquer publicação. No caso de expansão futura do sistema para acomodar outras sub-classes de Publicação e de Leitor, a mensagem acima continuará a ser usada sem necessidade de se alterar os métodos já implementados. Apenas novos métodos específicos precisarão ser escritos para as novas classes. Da mesma forma, caso se deseje modificar o privilégio de uma determinada classe de leitores já existente, a modificação só precisará ser feita no método da classe, sem afetar rotinas já escritas em outras classes. Essa característica contribui para aumentar a manutenibilidade e extensibilidade das aplicações.

5. CONCLUSÕES

O objetivo da construção deste protótipo foi o de investigar novas abordagens de desenvolvimento de sistemas de informação, utilizando o paradigma de orientação a objetos, em um ambiente apropriado. As conclusões principais que se podem retirar dessa experiência são as seguintes:

a. É viável construir sistemas com razoável grau de complexidade, com grande redução de investimento em projeto e programação, pela utilização das classes pré-definidads de uso geral já fornecidas pelo ambiente. Outras classes especializadas para uso em sistemas de informação poderão ser ainda idealizadas, aumentaando ainda mais a produtividade conseguida.

b. O ambiente de desenvolvimento favorece o desenvolvimento incremental e evolutivo de aplicações, pela facilidade em implementar modificações localizadas, e pela concisão do código produzido. É necessário, no entanto, que

sejam desenvolvidas metodologias que suportem o desenvolvimento disciplinado de aplicações de maior porte.

c. A utilização de uma linguagem orientada para objetos, com o suporte de uma grande biblioteca de classes, estimula a criação de soluções não normalmente encontradas em sistemas convencionais. Essas soluções podem alcançar níveis mais altos de sofisticação, pela disponibilidade de uma linguagem com maior riqueza de alternativas.

REFERÊNCIAS BIBLIOGRÁFICAS:

1. B.J. COX, Object-Oriented Programming, Reading, MA, Addison-Wesley, 1986, 274pp.

2. A. GOLDBERG e D. ROBSON, Smalltalk-80 The Language and its Implementation, Reading, MA, Addison-Wesley, 1983, 714pp.

3. DIGITALK INC., Smalltalk/V 286 Tutorial and Programming Handbook, Los Angeles, Digitalk, 1988, 561pp.

4. D. MAIER, J. STEIN, A. OTIS e A. PURDY, Development of an Object-Oriented DBMS, ACM SIGPLAN Notices, 21 (11), 472-482, Nov. 1986.

5. D.H.H. INGALLS, A Simple Technique for Handling Multiple Polymorphism, ACM SIGPLAN Notices, 21 (11), 347-349, Nov. 1986.

6. B. STROUSTRUP, The C++ Programming Language, Menlo Park (CA), Addison-Wesley, 1986.BORLAND INTERNATIONAL, INC.,

7. BORLAND INTERNATIONAL, INC. Turbo-Pascal 5.5 - Object-Oriented Programming Guide, Scotts Valley (CA), Borland, 1989.

8. B. MEYER, Object-Oriented Software Construction, Hemel Hempstead, Prentice Hall International (UK), 1988, 534pp.